Deciding Equivalence of Top-Down XML Transformations in Polynomial Time

Sebastian Maneth
National ICT Australia *
and University of New South Wales
Sydney, Australia
sebastian.maneth@nicta.com.au

Helmut Seidl
Technical University Munich
Munich, Germany
seidl@inf.tum.de

ABSTRACT
Many useful XML transformations can be formulated through deterministic top-down tree transducers. A canonical form for such transducers is presented which allows to decide equivalence of their induced transformations in polynomial time. If the transducer is total, the canonical form can be obtained in polynomial time as well.

Keywords
Xml, top-down tree transducer, equivalence, minimization.

1. INTRODUCTION
The transformation of Xml documents is of fundamental importance for practical Xml processing. Transformations are needed, e.g., for insertion of derived formating information or for adaptation of documents to the particular syntax demanded by a given application. Many routine Xml-transformations are *simple*, i.e., can be produced by a single top-down traversal over the tree structure underlying the input document. Such transformations include simple filterings, relabelings, insertions, and deletions as well as duplications of elements. Simple transformations can conveniently be formulated by means of *deterministic top-down tree transducers* running over a ranked-tree encoding of the given input document. An example of a top-down Xml transformation is shown in Figure 1; it copies the input document and additionally constructs a table of contents containing the titles $t_1, \ldots, t_n$ of all sections. A top-down tree transducer is a simple functional program: functions recursively generate trees through pattern matching on their single input tree argument.

Here we consider the problem of deciding whether or not two such transducers induce the same transformation. By an old result of Ésik, this problem is known to be decidable [7] (see also [4]). The involved algorithm, however, is based on upper bounds in the difference of sizes of intermediate trees appearing in derivations of the transducers. Since the algorithm explicitly keeps track of very large "difference trees", it seems hard to extract an efficient implementation. Instead, we introduce a new normal form for deterministic top-down tree transducers into which every transducer can be transformed. We prove that this normal form is *canonical* meaning that it only depends on the translation itself. Our canonical form can be seen as the generalization of a corresponding normal form for deterministic finite-state word-to-word transducers as considered by Mohri [12]. The canonicity of our normal form for deterministic top-down tree transducers allows for a kind of *minimization* of transducers — given that they satisfy the technical properties of being *uniform* and *earliest*. The latter property implies that a transducer produces its output as early as possible during a derivation. While the canonical form can be achieved for every deterministic top-down tree transducer, we show that it can be obtained even in polynomial time for total transducers, i.e., transducers whose transformation is defined for every input tree.

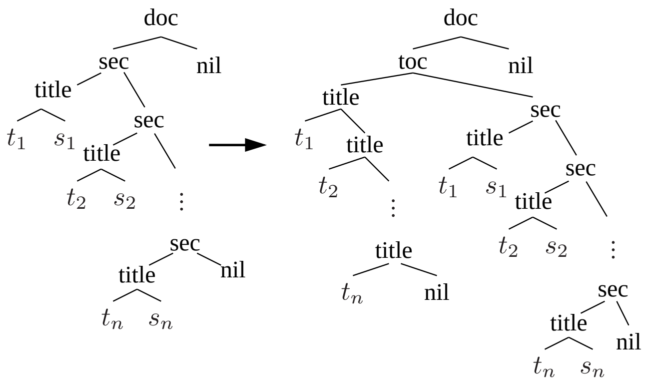

Figure 1: A typical top-down XML transformation.

Once the transducers in question are in the canonical form, deciding their equivalence becomes trivial: it boils down to checking isomorphism of their syntactical representations. These methods can also be extended to provide a practical procedure for deciding equivalence of deterministic top-down tree transducers with regular look-ahead. Such transducers additionally allow to test input trees for inclusion

*National ICT Australia is funded through the Australian Governments *Backing Australia's Ability initiative*, in part through the Australian Research Council.

in arbitrary regular tree languages. For practical purposes, such as query evaluation of XPath, this is a very useful property as it allows to check for the existence of (bottom-up) tree patterns in the input. Note also that every deterministic *bottom-up* tree transducer can be transformed into an equivalent deterministic top-down tree transducer with look-ahead [3]. Finally, note that for nondeterministic top-down tree transudcers the equivalence problem is undecidable, because this already holds for ε-free (one-way) finite-state *string* transducers [8].

The XPath query language is a popular formalism for selecting nodes from an XML document. A wide range of query and transformation languages, such as XQuery and XSLT, use XPath as their node selection formalism. An XPath expression is similar to a regular expression and is evaluated on the paths of the Xml tree, starting at the root node. The containment and equivalence problems are already coNP complete for a small fragment of XPath which only uses child, descendant, wildcard, and filter (branching) [10]. In the absence of any one of the operations descendant, wildcard, or filter containment is in PTIME [15, 2, 11]. It is possible to formulate an XPath query through a tree transducer: every input node is copied to the output, and a new unary "select symbol" is inserted above each node selected by the query. However, even simple queries such as "select all a-nodes that have a b-node descendant" cannot be realized by a top-down transducer in this way (because the transducer does not know of the presence of b-node descendants upon visiting an a-node). To remedy this problem, one can first relabel the input tree by the run of a tree automaton, or, equivalently, add regular look-ahead. Top-down tree transducers with regular look-ahead (which can be tested for equivalence using our methods) can indeed realize the above mentioned fragment of XPath. Note, however, that the use of nested filters in an XPath query is similar to a conjunction and will cause the look-ahead tree automaton to be of exponential size in the size of the query.

2. PRELIMINARIES

Top-down tree transducers work on ranked trees. This means that the number of the children of a node is determined by the *rank* of the symbol at that node. We therefore consider ranked alphabets Σ consisting of finitely many symbols where each symbol $a \in \Sigma$ is implicitly equipped with a rank $\rho(a) \in I\!N_0$, where rank 0 indicates that a is the potential label of a leaf.

The set $\mathcal{T}_\Sigma$ of ranked trees over the ranked alphabet Σ then is defined by

$$t ::= a(t_1, \ldots, t_k)$$

where a ranges over symbols in Σ of rank k. As usual, we also write a for the tree $a()$. We represent the nodes of a tree in Dewey notation, i.e., by sequences of numbers (for readability, numbers in the sequence are separated using dots). Formally, the set $V(t)$ of *nodes* of the tree t is inductively defined as: $V(t) = \{\varepsilon\}$ if $t = a()$, $a \in \Sigma$ of rank zero, and $V(t) = \{i.v \mid 1 \le i \le k, v \in V(t_i)\}$ if $t = b(t_1, \ldots, t_k)$, $b \in \Sigma$ of rank $k \ge 1$ and $t_1, \ldots, t_k \in \mathcal{T}_\Sigma$. Instead of $w.\varepsilon$ we simply write w. Thus, ε represents the root node and $u.i$ represents the ith child of the node u. The size of t, denoted size(t) is its number $|V(t)|$ of nodes.

A *pattern* p is the upper portion of a tree. Formally the set of all patterns is given by the set of all trees in $\mathcal{T}_{\Sigma\cup\{\top\}}$, where $\top$ is a new symbol of rank zero which is not in Σ. Assume p is a pattern containing exactly k occurrences of $\top$, and $t_1, \ldots, t_k$ is a sequence of patterns. Then the pattern $t = p[t_1, \ldots, t_k]$ is obtained from p by replacing the i-th occurrence of $\top$ (in left-to-right order) with t_i. Note that the result t is a tree, i.e., does not contain occurrences of $\top$ iff the $t_1, \ldots, t_k$ are all trees.

Consider the set $\mathcal{P}_\Sigma = \mathcal{T}_{\Sigma\cup\{\top\}} \cup \{\perp\}$ of all patterns enhanced with an extra element $\perp$ (not in $\Sigma \cup \{\top\}$). Then this set forms a complete lattice where the ordering is given by $\perp \sqsubseteq p$ for all p, and $p \sqsubseteq p'$ for patterns p, p' iff $p = p'[p_1, \ldots, p_k]$ for suitable patterns $p_1, \ldots, p_k$.

For a nonempty set $\mathcal{P}$ of patterns, its least upper bound $\sqcup\mathcal{P}$ is the unique pattern p such that p has a node v labeled by $a \in \Sigma$ iff v occurs in every pattern p' from $\mathcal{P}$ with the same label a.

While the lengths of strictly decreasing chains in $\mathcal{P}_\Sigma$ can be infinite, the lengths of strictly *increasing* chains are always finite. More precisely, the number of elements in a strictly increasing chain above a pattern p is bounded by the number $|p|_\Sigma$ of nodes in p which are labeled with elements from Σ.

3. TOP-DOWN DETERMINISTIC TREE TRANSDUCERS

A *top-down deterministic tree transducer* (transducer for short) is a tuple $T = (Q, \Sigma, \Delta, \delta, A)$, where

- Q is a finite set of states
- Σ and Δ are ranked input and output alphabets, respectively
- δ is the (possibly partial) transition function, and
- A is the initial expression.

The initial expression A has the form $p'[q'_1(x_0), \ldots, q'_m(x_0)]$ for a variable x_0 meant to be bound to the input tree, a pattern $p' \in \mathcal{T}_{\Delta\cup\{\top\}}$, and a sequence $q'_1, \ldots, q'_m$, $m \ge 0$ of states in Q.

For every state q in Q and input symbol $a \in \Sigma$ of rank k the transition function δ contains at most one transition, which is of the form

$$q(a(x_1, \ldots, x_k)) \to p[q_1(x_{i_1}), \ldots, q_r(x_{i_r})]$$

where $x_1, \ldots, x_k$ are distinct variables, $p \in \mathcal{T}_{\Delta\cup\{\top\}}$ is a pattern, $q_1, \ldots, q_r \in Q$, and x_{i_j} are variables occurring among the $x_1, \ldots, x_k$. For every state q and input symbol a let $\delta(q, a)$ be the right-hand side of the transition for q and a if it is defined, and let $\delta(q, a)$ be undefined otherwise. The transducer is *total* iff $\delta(q, a)$ is defined for all $q \in Q$ and $a \in \Sigma$.

Every state of the transducer can be considered as a unary function which recurses over its argument. Assume the argument s of q is of the form $s = a(s_1, \ldots, s_k)$. Assume further that $p[q_1(x_{i_1}),\ldots,q_r(x_{i_r})]$ is the right-hand side in δ for q and the symbol a. If the recursive calls $q_j(s_{i_j})$ return results t_j, then the call $q(s)$ returns the value $[\![q]\!](s) = p[t_1,\ldots,t_r]$. If on the other hand, there is no transition in δ for q and a or one of the recursive calls $q_j(s_{i_j})$ is undefined, then the function $[\![q]\!]$ is also undefined for s.

Now assume that the initial expression A is given by $A = p'[q'_1(x_0),\ldots,q'_m(x_0)]$. Then the output $T(s)$ of the transducer T on input s is given by:

$$T(s) = p'[[\![q'_1]\!](s),\ldots,[\![q'_m]\!](s)]$$

if $[\![q'_j]\!](s)$ is defined for all $1 \leq j \leq m$.

In particular, we call two transducers T_1 and T_2 *equivalent* iff their domains coincide and $T_1(s) = T_2(s)$ whenever T_i are defined for the input tree s.

Example 1 We want to define a transducer that induces the translation of XML documents with section and title markup as described in the Introduction. The transducer has $A = q_0(x_0)$, states q_0, t, e, and id, and consists of the following transitions.

$$\begin{array}{lll} q_0(\mathrm{doc}(x_1,x_2)) & \to & \mathrm{doc}(\mathrm{toc}(t(x_1),\mathrm{id}(x_1)),\mathrm{nil}) \\ t(\mathrm{sec}(x_1,x_2)) & \to & \mathrm{title}(e(x_1),t(x_2)) \\ t(\mathrm{nil}) & \to & \mathrm{nil} \\ e(\mathrm{title}(x_1,x_2)) & \to & \mathrm{id}(x_1) \end{array}$$

The state id has the obvious transitions in order to realize identity. Note that the right-hand side of the first transition equals $p[t(x_1),\mathrm{id}(x_1)]$ with pattern $p = \mathrm{doc}(\mathrm{toc}(\top,\top),\mathrm{nil})$.

Wildcards

Note that query languages such as XPATH support a wildcard operator for selecting a node with *any* label. Such a mechanism for dealing with arbitrary labels is also present in pattern matching constructs of mainstream programming languages in form of the "default case". For a fixed, finite set of ranks, this can be obtained in our setting by enhancing the input alphabet Σ with special symbols "$*_k$", representing input labels of rank k that are arbitrary, but not in Σ. Then, a transition of the form $q(*_k(x_1,\ldots,x_k)) \to *_k(q(x_1),\ldots,q(x_k))$ copies any non-Σ symbol from the input into the output tree. Note that, in the context of XML we typically work on binary trees (with leaves representing the empty hedge) and henceforth only need one incarnation of the $*$-symbol of rank two.

4. COMMON PATTERNS

Consider a state q of the transducer T whose translation is nonempty. Consider the pattern

$$\mathsf{pref}(q) = \bigsqcup\{[\![q]\!](s) \mid s \in \mathcal{T}_\Sigma \text{ and } [\![q]\!](s) \text{ is defined}\}.$$

Since the set of patterns is a complete lattice, the pattern $\mathsf{pref}(q)$ is well-defined. Intuitively, $\mathsf{pref}(q)$ is the prefix tree that is common to all outputs that can be generated by the state q. For instance, consider the transducer M_1 which has the following two transitions:

$$\begin{array}{lll} q(a(x_1,x_2)) & \to & d(q(x_1),d(q(x_1),e)) \\ q(e) & \to & d(d(e,e),d(e,e)) \end{array}$$

Obviously, all outputs generated by the state q start with the prefix tree $d(\top,d(\top,\top))$. In fact, the largest prefix tree of q is the pattern $\mathsf{pref}(q) = d(d(\top,\top),d(\top,e))$.

In the following we show how to compute common prefixes. In order to do so we consider *uniform* transducers. Let $\mathcal{C}$ denote the largest collection $\mathcal{C}'$ of sets of T such that $Q' \in \mathcal{C}'$ implies that for every $a \in \Sigma$, one of the following statements hold:

1. Either there is no transition for a state $q' \in Q'$ and a, or
2. For every $q' \in Q'$, there is a transition $q'(a(x_1\ldots,x_k)) \to t_{q',a}$ where, additionally, each set $Q'_j = \{q'' \mid \exists\, q' \in Q' : q''(x_j) \text{ occurs in } t_{q',a}\}$ $(j = 1,\ldots,k)$ is again in C.

We call a set Q' of states of T *consistent* iff $Q' \in \mathcal{C}$. Note that, according to this definition, the empty set of states is always consistent. Also, if Q' is consistent then so is every subset of Q'. Given this definition of consistency, we call the top-down deterministic tree transducer T with a nonempty transformation *uniform* iff the following hold:

1. For every state q, the domain of the partial function $[\![q]\!]$ is nonempty;
2. The set of states occurring in the initial expression is consistent;
3. For every right-hand side t of a rule and every j, the set of states q for which $q(x_j)$ occurs in t is consistent.

For a *total* transducer, $q_1 \sim q_2$ for all states q_1, q_2, because $\delta(q,a)$ is defined for all states q and input symbols a. Therefore, every total transducer is in particular uniform. Note, however, that the transducer from Example 1 is not total but still uniform.

We now show that *every* transducer is (effectively) equivalent to a uniform transducer.

Lemma 2 *For every transducer T with a nonempty domain, a uniform transducer T' can be constructed in exponential time such that T and T' are equivalent.*

Proof. Assume $T = (Q,\Sigma,\Delta,\delta,A)$. Let us call a set $B \subseteq Q$ *inhabitated* iff the intersection of the domains of the functions $[\![q]\!], q \in B$, is nonempty.

The idea for the new transducer T' is to extend each state of T with an extra component $B \subseteq Q$ which records the set of all other states currently running in parallel on the same portion of the input tree. Thus, the states of T' are of the form $\langle q,B\rangle, q \in B \subseteq Q$, where we take care that B is inhabitated. If $A = p'[q'_1(x_0),\ldots,q'_m(x_0)]$, then the initial expression A' of T' is given by

$$A' = p'[\langle q'_1,B\rangle(x_0),\ldots,\langle q'_m,B\rangle(x_0)]$$

where $B = \{q'_1, \ldots, q'_m\}$ and $\langle q'_1, B\rangle, \ldots, \langle q'_m, B\rangle$ are new states of T'. Since the domain of T is nonempty, the set B is inhabitated.

For a new state $\langle q, B\rangle$ and an input symbol $a \in \Sigma$, the transducer T' has states $\langle q_1, B_1\rangle, \ldots, \langle q_r, B_r\rangle$ and the transition

$$\langle q, B\rangle(a(x_1, \ldots, x_k)) \rightarrow p[\langle q_1, B_1\rangle(x_{i_1}), \ldots \langle q_r, B_r\rangle(x_{i_r})]$$

if the following conditions are satisfied:

(1) $\delta(q, a) = p[q_1(x_{i_1}), \ldots q_r(x_{i_r})]$

(2) for all $q' \in B$, $\delta(q', a)$ is defined, and

(3) for every $1 \leq j \leq r$, B_j is inhabitated and consists of all states q''_l for which there is a transition

$$q''(a(x_1, \ldots, x_k)) \rightarrow p''[q''_1(x_{i_1}), \ldots q''_{r''}(x_{i_{r''}})]$$

in δ (for the same symbol a) with $q'' \in B$ and $x_{i_l} = x_{i_j}$.

It is readily verified that:

1. every set $\{\langle q, B\rangle \mid q \in B\}$ of states $\langle q, B\rangle$ constructed by the algorithm is consistent;
2. $[\![\langle q, B\rangle]\!](s) = [\![q]\!](s)$ whenever $[\![\langle q, B\rangle]\!](s)$ is defined.

Thus, by construction of the initial expression A' of T', T and T' are equivalent. Moreover, the domains of all states $\langle q''_j, B_j\rangle$ which occur in the same right-hand side applied to the same variable agree in their second components. Since all occurring sets B are consistent by construction, we conclude that T' must be uniform.

Checking a set of states B for being inhabitated, can be done in exponential time, by checking emptiness of a corresponding alternating top-down tree automaton. If T has n states, then the number of states of the resulting transducer T' is at most $n \cdot 2^{n-1}$. Thus, it can be verified that the whole construction can be performed in deterministic exponential time. $\diamond$

The number of states constructed for T' in the proof of Lemma 2 heavily depends on the number (of combinations) of different states that translate a node in parallel. In practice we expect that this number is not large. In fact, the bulk of practical translations are of *linear size increase*, i.e., the size of every output tree is bounded by a constant times the size of the corresponding input tree. It is well-known by an old result of Aho and Ullman [1] that for any linear size increase (deterministic) top-down tree translation there is (effectively) a transducer that is "finite-copying". The latter means that the number of states translating any input node is bounded by a constant c (called the "copying number"). Clearly, for a transducer with copying number c, the running time of the construction of Lemma 2 is at most exponential in c. Note that the transducer of Example 1 has copying number 2.

For a uniform transducer T, let $\eta(T)$ denote the maximal size of a minimal output tree produced for state of T, i.e., $\eta(T) = \max\{\eta_q \mid q \in Q\}$ where, for $q \in Q$,

$$\eta_q = \min\{\text{size}([\![q]\!](s)) \mid s \text{ in domain of } [\![q]\!]\}.$$

Then we have:

Theorem 3 *Assume $T = (Q, \Sigma, \Delta, \delta, A)$ is a uniform transducer. The common patterns $\mathsf{pref}(q)$, $q \in Q$, can be computed in time $\mathcal{O}(|T| \cdot \eta(T)^2)$.*

Proof. We construct the following system of in-equations for the sets of unknown patterns $Y_q, q \in Q$:

$$Y_q \sqsupseteq p[Y_{q_1}, \ldots, Y_{q_r}]$$

for every transition

$$q(a(x_1, \ldots, x_k)) \rightarrow p[q_1(x_{i_1}), \ldots, q_r(x_{i_r})]$$

in δ. Here, we assume substitution into the pattern p as a *strict* operation, meaning that $p[p_1, \ldots, p_r] = \perp$ already when $p_i = \perp$ for some i. Obviously, each right-hand side in this constraint system is monotonic in its arguments. A closer look, however, reveals that it is even distributive for non-$\perp$ arguments, meaning that for any nonempty set $\mathcal{S}$ of sequences $(p_1, \ldots, p_r)$ with $p_j \neq \perp$ and least upper bound $(\bar{p}_1, \ldots, \bar{p}_r)$,

$$p[\bar{p}_1, \ldots, \bar{p}_r] \;=\; \bigsqcup\{p[p_1, \ldots, p_r] \mid (p_1, \ldots, p_r) \in \mathcal{S}\}.$$

Let $y_q^{(j)}$ denote the value of the variable Y_q after j rounds of fixpoint iteration. By induction on j, we prove that

$$y_q^{(j)} \;=\; \bigsqcup\{[\![q]\!](s) \mid \mathsf{depth}(s) \leq j, [\![q]\!](s) \text{ is defined}\}.$$

Note that in the proof of the induction step, both assumptions on uniformity as well as on distributivity are crucial.

We conclude that for all states q, the least upper bound to the $y_q^{(j)}, j \geq 0$, equals the pattern $\mathsf{pref}(q)$. Since the lengths of strictly increasing chains in the pattern lattice is finite, fixpoint iteration will indeed terminate after a finite number of steps. In fact, the only non-$\perp$ values attained by Y_q during fixpoint computation are patterns p exceeding t_q, where t_q is the smallest output tree $[\![q]\!](s)$ for any s in the domain of $[\![q]\!](s)$. Thus, the maximal number of updates to each fixpoint variable is bounded by 1 + the size of the tree t_q. Moreover, each occurring value p for Y_q can be represented in space $\mathcal{O}(\eta(T))$. Thus, it is easy to see that the whole fixpoint computation can be executed within the given time bound. $\diamond$

Consider a total transducer T. For every state q and every symbol e of rank 0, T has a transition $q(e) \rightarrow t_{q,e}$ for some tree $t_{q,e} \in \mathcal{T}_\Delta$. A rough upper bound to the sizes of such trees is given by the size of T itself. Hence, according to Theorem 3, the common patterns for all states can be computed for total transducers in cubic time.

In the case of uniform transducers, however, we do not have at hand the small trees $t_{q,e}$ as for total transducers. Instead, however, we can rely for a state q, on *some* output tree t_q returned by $[\![q]\!]$. Such a tree exists of depth bounded by the size of T. Accordingly the size of this tree is at most exponential in the size of T. Hence in this case, the value $\eta(T)$ of Theorem 3 can be at most be *exponential*.

As an example, consider the transducer M_1 of before which has these two rules

$$\begin{array}{lll} q(a(x_1,x_2)) & \rightarrow & d(q(x_1), d(q(x_1), e)) \\ q(e) & \rightarrow & d(d(e,e), d(e,e)). \end{array}$$

Clearly, fixpoint iteration will terminate after only two steps:

$$\begin{array}{lll} y_q^1 & = & d(d(e,e)d(e,e)) \\ y_q^2 & = & d(d(\top,\top), d(\top,e)) = y_q^3. \end{array}$$

5. EARLIEST TRANSDUCERS

A uniform transducer T is called *earliest* iff $\mathsf{pref}(q) = \top$ for all states q of T.

Theorem 4 *Every transducer T is effectively equivalent to an earliest transducer T'. If T is uniform, then T' can be constructed in time $\mathcal{O}(|T| \cdot \eta(T)^2)$.*

Proof. By Lemma 2, we can construct for every transducer an equivalent uniform transducer.

Therefore, assume that the transducer T is uniform. By Theorem 3, we can compute for every state q of T, the pattern $\mathsf{pref}(q)$ which is common to all outputs produced by q. The idea then is to produce this common pattern as early as possible. Together with the state q, we additionally record the node v in the pattern $\mathsf{pref}(q)$ which is to be expanded next. This means the set of states of the new transducer T' consists of pairs $\langle q, v\rangle$ where $q \in Q$, and v is one of the nodes of $\mathsf{pref}(q)$ labeled with $\top$.

If $A = p'[q_1'(x_0), \ldots, q_m'(x_0)]$ is the initial expression of T, then the initial expression A' of T' is given by:

$$\begin{array}{lll} A' & = & p'[\; p_1'[\langle q_1', v_{1,1}\rangle(x_0), \ldots, \langle q_1', v_{1,l_1}\rangle(x_0)], \ldots, \\ & & \quad p_m'[\langle q_m', v_{m,1}\rangle(x_0), \ldots, \langle q_m', v_{m,l_m}\rangle(x_0)] \;] \end{array}$$

where $v_{j,1}, \ldots, v_{j,l_j}$ is the left-to-right sequence of nodes in $p_j = \mathsf{pref}(q_j')$ labeled with $\top$.

We now construct the transitions of the transducer T'. Consider a state $\langle q, v\rangle$ and an input symbol a of rank k, and assume that T has the transition:

$$q(a(x_1, \ldots, x_k)) \rightarrow p[q_1(x_{i_1}), \ldots, q_r(x_{i_r})].$$

Then, we may assume that $\mathsf{pref}(q)$ and p are compatible, i.e., the greatest lower bound $\mathsf{pref}(q) \sqcap p$ is a pattern (and thus different from $\perp$).

We distinguish two cases:

Case 1: $v = v'.u$ such that v' is the j-th occurrence of $\top$ in p.

This means that the pattern that is common to all outputs generated by q spans over the entire path v' in p (and possibly further into another right-hand side, viz. u nonempty).

Then let p' denote the sub-pattern of $\mathsf{pref}(q)$ at node v', and $p_j = \mathsf{pref}(q_j)$. Then $p_j \sqsubseteq p'$ should hold implying that the remainder u of v is a node in p_j. Then let p'' denote the sub-pattern of p_j at node u, with $\top$-leaves at nodes $v_1, \ldots, v_l$. In this case, we generate the transition:

$$\begin{array}{ll} \langle q, v\rangle(a(x_1, \ldots, x_k)) \rightarrow p''[& \langle q_j, u.v_1\rangle(x_{i_j}), \ldots, \\ & \langle q_j, u.v_l\rangle(x_{i_j}) \;] \end{array}$$

Case 2: v is a node of p. This means that the path v is fully contained in p, i.e., ends in p.

Let $p'[q_f(x_{i_f}), \ldots, q_{f+l}(x_{i_{f+l}})]$ denote the subexpression of the right-hand side of the transition for q and a at node v, and let $p_j = \mathsf{pref}(q_j)$ for $j = f, f+1, \ldots, f+l$. Then we generate the new transition:

$$\langle q, v\rangle(a(x_1, \ldots, x_k)) \quad \rightarrow \quad p'[A_f, \ldots, A_{f+l}]$$

where, for $j = f, f+1, \ldots, f+l$,

$$A_j \;=\; p_j[\langle q_j, v_{j,1}\rangle(x_{i_j}), \ldots, \langle q_j, v_{j,l_j}\rangle(x_{i_j})]$$

and $v_{j,1}, \ldots, v_{j,l_j}$ is the left-to-right sequence of nodes in p_j that are labeled $\top$.

It should be clear that T' is indeed earliest: a state $\langle q, v\rangle$ implies that node v in $\mathsf{pref}(q)$ is labeled $\top$. By the definition of pref, this means that there are input trees s_1 and s_2 such that the label of v in $[\![q]\!](s_1)$ is different from the label of v in $[\![q]\!](s_2)$; hence, by point 2 of the claim below, there are s_1', s_2' such that the root node of $[\![\langle q, v\rangle]\!](s_1')$ is labeled different than the root node of $[\![\langle q, v\rangle]\!](s_2')$, i.e., $\mathsf{pref}(\langle q, v\rangle) = \top$. The following claim suffices in order to prove the correctness of the construction.

Claim: Assume q is a state of T with $p = \mathsf{pref}(q)$ where v is a leaf in p labeled with $\top$. Then for every input tree s the following holds:

1. $[\![q]\!](s)$ is defined iff $[\![\langle q, v\rangle]\!](s)$ is defined.
2. If $[\![q]\!](s)$ is defined then $[\![\langle q, v\rangle]\!](s)$ equals the subtree of $[\![q]\!](s)$ at node v.

The claim follows again by structural induction on s. $\diamond$

Again, the complexity of the construction crucially depends on the structural parameter $\eta(T)$. In particular for total transducers, the corresponding earliest transducer can be constructed in cubic time.

As an example, consider a transducer T with initial expression $A = q(x_0)$ and states q, q_1, q_2. Assume that

$$\begin{array}{lll} \mathsf{pref}(q) & = & b(b(\top, b(\top,\top), b(b(\top,\top),\top))) \\ \mathsf{pref}(q_2) & = & b(b(\top,\top), a(\top,\top)) \end{array}$$

and that there is a transition

$$q(a(x_1, x_2) \rightarrow b(b(e, b(b(q_1(x_1), e), e)), q_2(x_2)).$$

The initial expression of the new transducer T' is

$$\begin{array}{l} A' = b(b(\langle q, 1.1\rangle(x_0), b(\langle q, 1.2.1\rangle(x_0), \langle q, 1.2.2\rangle(x_0)), \\ \qquad b(b(\langle q, 2.1.1\rangle(x_0), \langle q, 2.1.2\rangle(x_0)), \langle q, 2.2\rangle(x_0)))). \end{array}$$

Let us now construct the transition for $\langle q, 2.1.1\rangle$ and the input symbol a. This is Case 1 in the proof of Theorem 4 with $v' = 2$ and $u = 1.1$. Since $\mathsf{pref}(q_2)$ exactly generates the two b-nodes that are missing from $\mathsf{pref}(q)$, the pattern of this transition is simply $\top$ and we obtain as transition

$$\langle q, 2.1.1\rangle(a(x_1, x_2)) \to \top[\langle q_2, 1.1\rangle(x_2)] = \langle q_2, 1.1\rangle(x_2).$$

6. MINIMIZING EARLIEST TRANSDUCERS

The key property of earliest transducers is that they produce the respective output trees in a canonical fashion. This means that for two states q_1 and q_2 of an earliest transducer T, the (partial) functions $[\![q_1]\!]$ and $[\![q_2]\!]$ agree if and only if the patterns on the right-hand sides of all corresponding transitions agree as well as the corresponding recursive calls in the right-hand sides.

Formally, let T denote an earliest transducer. Recall that by definition T then is also deterministic and uniform. On the set Q of states of T we establish an equivalence relation $\equiv$ as the *largest* equivalence relation $\equiv'$ such that $q \equiv' q'$ implies the following:

If there is a transition

$$q(a(x_1, \ldots, x_k)) \to p[q_1(x_{i_1}), \ldots, q_r(x_{i_r})],$$

then there also exists a transition

$$q'(a(x_1, \ldots, x_k)) \to p'[q'_1(x_{i'_1}), \ldots, q'_{r'}(x_{i'_{r'}})]$$

for the same symbol a such that $p = p'$, $r = r'$, and for all $j = 1, \ldots, r$, $x_{i_j} = x_{i'_j}$ and $q_j \equiv' q'_j$.

Theorem 5 *Assume T is an earliest transducer. Then the following holds:*

1. $q \equiv q'$ *iff* $[\![q]\!] = [\![q']\!]$.
2. *The equivalence relation $\equiv$ can be computed in polynomial time.*
3. *The unique minimal earliest transducer T' equivalent to T can be constructed in polynomial time.*

We are ready to state the main result of this paper.

Theorem 6 *The equivalence of top-down deterministic tree transducers T_1, T_2 is decidable. If the transducers are total or earliest, then equivalence can be decided in polynomial time. If the transducers are uniform, then equivalence is at most in PSPACE.*

Proof. The result for total and earliest transducers follows from Theorem 5. This theorem gives us an upper complexity bound also for general uniform transducers. If however, one of the structural parameters $\eta(T_i)$ is exponential, the corresponding algorithm also runs in exponential time.

A PSPACE algorithm for deciding equivalence of uniform transducers, however, can be constructed as follows. It successively constructs a path in the output tree where the two input transducers differ. In order to produce this path, at most two paths in the input tree must be tracked. Since in case of equivalence, any transducer can be only linearly ahead of the other on the path, the procedure either may stop when the delay on the path increases too much or a node is output where the two transducers differ. Since the information to be recorded is just polynomial, the upper PSPACE bound follows. $\diamond$

A useful extension of the top-down tree transducer is the top-down tree transducer with *regular look-ahead* [3]. Such a transducer can test its input trees for inclusion in arbitrary regular tree languages (by means of a bottom-up tree automaton called the "look-ahead automaton"). Alternatively, one can think of a transducer with look-ahead as the composition of two translations: first, the input tree is relabeled (each a-labeled node u of the input tree is relabeled by $\langle a, p_1, \ldots, p_k\rangle$ if the look-ahead automaton arrives in state p_i at the ith subtree of u). The second translation is an ordinary top-down tree transducer, running on the relabeled input tree. Clearly we can use the decision procedure for equivalence of top-down tree transducers [7] (or, alternatively, Theorem 6) to decide equivalence of deterministic top-down tree transducers with regular look-ahead: simply construct a relabeling that adds, for the look-ahead automata of both transducers, the current look-ahead state and the look-ahead states at all children nodes. The top-down tree transducers then merely have to be extended to additionally check that the input tree is a correct relabeling; this can easily be done by additionally keeping in the states of the transducers the states of the look-ahead automata at the current node.

Corollary 7 *The equivalence problem for deterministic top-down tree transducers with look-ahead is decidable.*

Clearly, Corollary 7 can be used to check whether or not two transducers (possibly with look-ahead) are equivalent on a given regular set R of input trees, by simply letting their look-ahead automata check containment in R.

7. OPEN PROBLEMS

In the context of XML there have been attempts to generalize top-down transducers to unranked trees, e.g. [9, 14, 13]. Such transducers cannot be simulated by ordinary top-down tree transducers on ranked-tree encodings, because they implicitly support concatenation of trees. Is equivalence of such transducers decidable? Can they be transformed into a normal form similar to the one presented here?

Another popular model of tree transducer is the macro tree transducer [6]. It can be seen as a generalization of top-down tree transducers by adding context-parameters to states. It is a long standing open problem whether or not equivalence for deterministic macro tree transducers is decidable. Recently it has been proved that equivalence is decidable for deterministic macro tree transductions that are of linear size increase [5], i.e., for which the size of every output tree is bounded by a constant times the size of the corresponding input tree. Note that this result is incomparable to Theorem 6: the methods from [5] do not help whenever the transducers produce output whose size is not linearly bounded by the size of the corresponding input.

8. REFERENCES

[1] A. V. Aho and J. D. Ullman. Translations on a context-free grammar. *Inform. and Control*, 19:439–475, 1971.

[2] S. Amer-Yahia, S. Cho, L. V. S. Lakshmanan, and D. Srivastava. Minimization of tree pattern queries. In *Proc. SIGMOD Conference*, pages 497–508, 2001.

[3] J. Engelfriet. Top-down tree transducers with regular look-ahead. *Math. Systems Theory*, 10:289–303, 1977.

[4] J. Engelfriet. Some open questions and recent results on tree transducers and tree languages. In R.V. Book, editor, *Formal language theory; perspectives and open problems*. Academic Press, New York, 1980.

[5] J. Engelfriet and S. Maneth. The equivalence problem for deterministic MSO tree transducers is decidable. *Inform. Proc. Letters*, 100:206–212, 2006.

[6] J. Engelfriet and H. Vogler. Macro tree transducers. *J. Comp. Syst. Sci.*, 31:71–146, 1985.

[7] Z. Ésik. Decidability results concerning tree transducers I. *Acta Cybernetica*, 5:1–20, 1980.

[8] T. V. Griffiths. The unsolvability of the equivalence problem for Λ-free nondeterministic generalized machines. *J. ACM*, 15:409–413, 1968.

[9] S. Maneth and F. Neven. Structured Document Transformations Based on XSL. In *Proc. DBPL*, pages 80–98, 1999.

[10] G. Miklau and D. Suciu. Containment and equivalence for a fragment of XPath. *J. ACM*, 51:2–45, 2004.

[11] T. Milo and D. Suciu. Index structures for path expressions. In *Proc. ICDT*, pages 277–295, 1999.

[12] M. Mohri. Minimization algorithms for sequential transducers. *Theor. Comput. Sci.*, 234:177–201, 2000.

[13] T. Perst and H. Seidl. Macro forest transducers. *Inf. Process. Lett.*, 89:141–149, 2004.

[14] A. Tozawa. Towards Static Type Inference for XSLT. In *ACM Symp. on Document Engineering*, pages 18–27, 2001.

[15] M. Yannakakis. Algorithms for acyclic database schemes. In *Proc. VLDB*, pages 82–94, 1981.